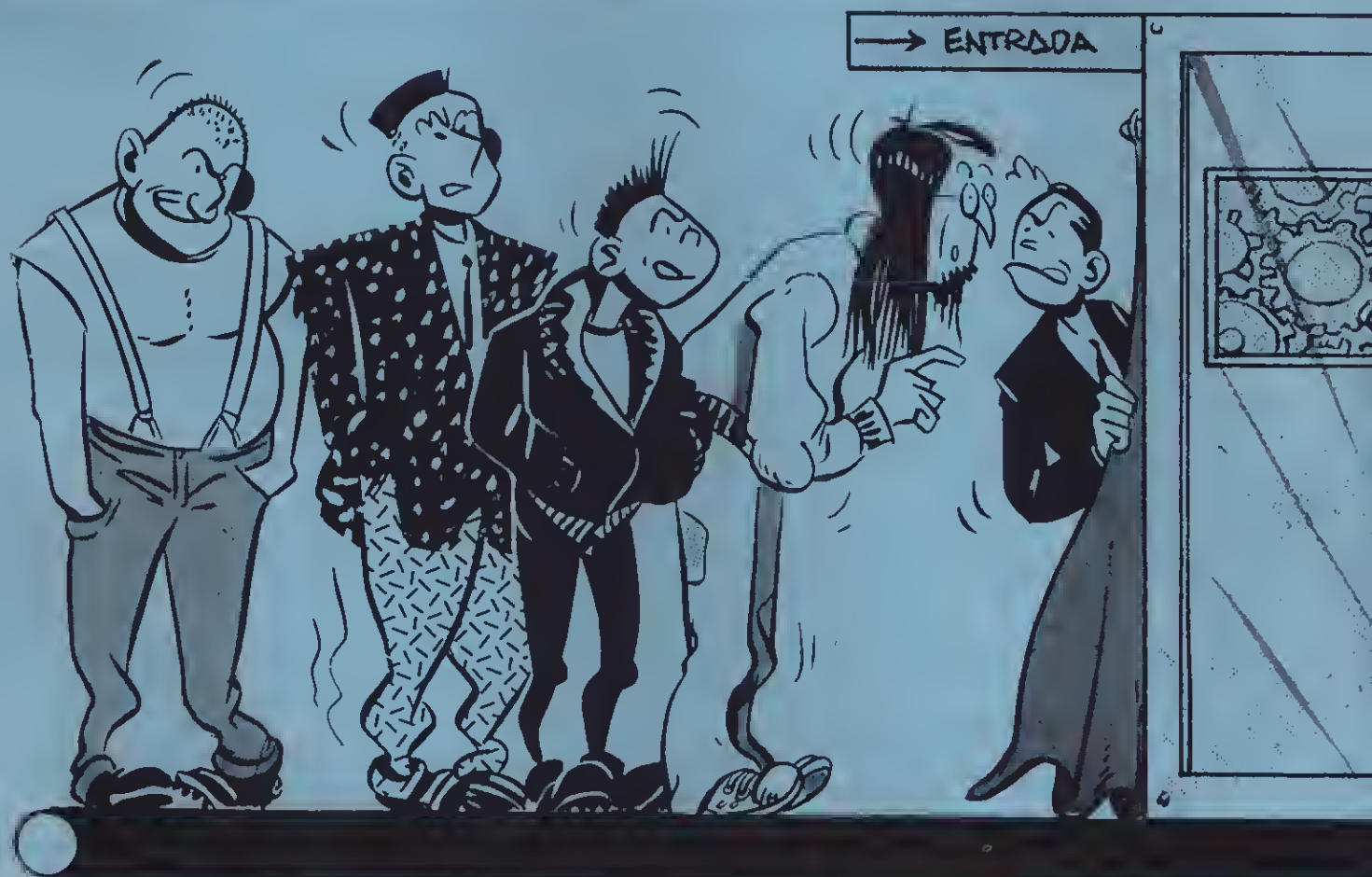


# RUTINAS DE CODIGO MAQUINA



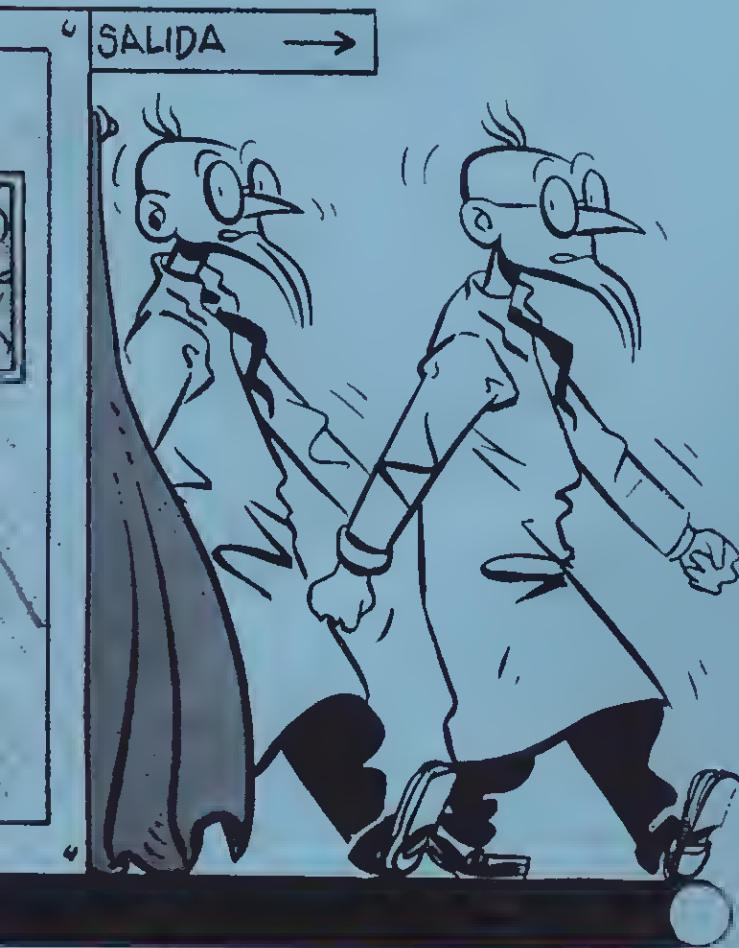
El BIOS (Basic Input Output System) se encuentra en la ROM de tu ordenador. Consiste en una serie de rutinas, escritas en código máquina, capaces de gestionar cosas tan dispares como el teclado, la pantalla, el interfaz de la impresora y el del cassette, los puertos de los joystick y las ranuras de los cartuchos.

Cualquiera que disponga de un desensamblador habrá comprobado que las posiciones de memoria más bajas de la ROM contienen una serie de saltos absolutos hacia diferentes direcciones (JP dirección). Quizá os hayáis preguntado por qué se desperdicia así tal cantidad de memoria (tree bytee

para cada rutina), ya que sería igual referirse a la posición final, en lugar de pasar por un salto absoluto. Pues bien, esto se hace en orden a asegurar totalmente la compatibilidad de los diferentes ordenadores MSX, así como de sus futuras mejoras y versiones. Microsoft, la firma creadora del estándar, dictó unas normas a seguir por todos los programadores, que deben ser estrictamente respetadas para que cualquier diferencia en el hardware no repercuta en el funcionamiento del software. Unos ejemplos aclararán mejor este punto. Supón que quieres escribir un dato en el cassette, poner en marcha el motor, encender el diodo de las mayúsculas o, simplemente, sacar un carácter por la pantalla. En



cualquiera de estos casos hay una forma directa de obtener el resultado de entrada/salida. No obstante, el mínimo cambio en la asignación comportaría que el ordenador mostrara unos resultados completamente inesperados.



Todo lo anterior conduce a la necesidad de **acceder a las rutinas del BIOS** en lugar de improvisar evoluciones de compromiso. Dicho esto, se aprecia claramente la importancia de contar con un mapa de la ROM que dé información de la ubicación y contenido de las rutinas fundamentales. A continuación se detallan, añadiendo, en las más interesantes, una relación de los parámetros de entrada necesarios en cada caso, así como de las modificaciones que efectúan en los registros y en las posiciones de memoria.

Sin duda encontrará inestimable la ayuda que os brindan las rutinas del BIOS. Desde aquí, el deseo de una fructífera programación.

## LAS RUTINAS DEL BIOS

### Posición: &H0

Esta rutina no necesita parámetros de entrada ni tampoco ofrece ninguno a la salida. Puede ser ejecutada utilizando un reetart (RST 0). Su función es la de inicializar el ordenador. Por consiguiente, se llama cuando se quiere empezar de nuevo, cuando se pulea el botón de reet o, automáticamente, al encender el aparato.

### Posición: &H8 y &H10

Estas rutinas son utilizadas por el intérprete BASIC para analizar los errores de sintaxis, tomar el siguiente carácter o token del programa, etc. Son de poca utilidad, aunque una posible aplicación sería la de construir un BASIC extendido.

### Posición: &HC

Se usa para leer una dirección de memoria de un cartucho determinado. El número de cartucho ha de colocarse en el acumulador y la dirección en el registro HL. Altera AF, BC y DE.

### Posición: &H14

Igual que la anterior pero para escribir.

### Posición: &H18

Es, sin duda, una rutina muy útil. Puede ser llamada con RST 18. Se encargará de sacar el carácter contenido en el acumulador al periférico seleccionado. Si la posición de memoria &HF416 contiene un cero, la salida será a la pantalla. Si &HF416 es distinto de cero, la salida será por impresora. Por último, tienes la posibilidad de escribir en un fichero de disco, cargando &HF864 con la dirección de memoria de dicho fichero, que señalará el dato a mandar. RST 18 no modifica ningún registro. Por otra parte, realiza una llamada al gancho situado en &HFEE4 después de guardar el par AF en la pila. Como puedes intuir, poner un parche en esa dirección te dará la oportunidad de controlar los distintos periféricos a tu antojo.

### Posición: &H1C

Esta rutina se emplea para ejecutar una subrutina de un cartucho.

### Posición: &H20

Puedes comparar los registros DE y HL llamando a esta rutina. Aquí tienes su listado:

LD	A, D
CP	H
RET	NZ
LD	A, E
CP	L
RET	

### Posición: &H24

Esta rutina selecciona una página de un cartucho.





### Posición &H28

Es empleada por el intérprete BASIC para conocer el tipo de variable que se está utilizando. Alternativamente se puede leer la dirección &HF663, puesto que siempre se almacena aquí el número de bytes de la variable usada; es decir, dos para las variables numéricas enteras, cuatro para las de precisión sencilla, ocho para las de doble precisión y tree para las cadenas alfanuméricas. Sin embargo, no es seguro que esta dirección se respete en futuras versiones. Por tanto observa si el flag C está a 0 (tipo 8), el flag M está a 1 (tipo 2), el flag Z está a 1 (tipo 3) o el flag P se encuentra a 0 (tipo 4).

### Posición &H30

Ejecuta una rutina contenida en un cartucho. El byte siguiente al RST 30 debe contener el identificador del cartucho y después debe colocarse la dirección de llamada.

### Posición &H38

Esta rutina es ejecutada 50 veces por segundo, salvo que las interrupciones estén desactivadas. Lo primero que hace es guardar los registros en la pila (incluidos los alternativos y los de índice), por lo que podrás emplearlos todos libremente y sin restricciones. Si pones un parche en la dirección &HFD9A forzarás al sistema operativo a ejecutar una de tus rutinas siempre que se produzca una interrupción. Como puedes ver, esto te da un poder inmenso sobre el ordenador. No modifica ningún registro, pero altera muchas posiciones de memoria, ya que actualiza, entre otras, la variable TIME y las escalas musicales. Asimismo, comprueba las colisiones de los SPRITES, el teclado, etc.

### Posición: &H41

Llamándola haces que la pantalla se desconecte. No obstante, todo lo que se escriba se conservará y podrás visualizarlo con la siguiente rutina. Suele ser útil cuando se hace un dibujo muy complicado que se quiere mostrar en pantalla instantáneamente. Modifica los pares AF y BC.

### Posición: &H44

Esta rutina activa la pantalla, por lo que complementa a la anterior. Al igual que aquella, modifica los registros AF y BC.

### Posición: &H47

Se llama a esta rutina para escribir en uno de los registros de estado del procesador de vídeo (VDP). En C debe ponerse el número de registro a escribir y en B el dato en cuestión. Su equivalente en BASIC sería: VDP(C)=B. Es importante emplear esta rutina, en lugar de acceder al VDP directamente, puesto que se encarga de guardar una copia del registro de estado en la RAM del sistema, desde la posición &HF3DF hasta la &HF3E6. Ten presente que estos registros sólo son de escritura y no podrías comprobar los datos una vez mandados. Modifica los

parees AF y BC.

### Posición &H4A

Funciona igual que la instrucción VPEEK del BASIC. Debes cargar la dirección de la RAM de vídeo en el par HL y obtendrás a la salida el resultado en el acumulador. Modifica sólo AF.

### Posición: &H4D

Es idéntica a la anterior sólo que ésta actúa como VPOKE. El dato a escribir ha de ponerse en el acumulador.

### Posición: &H50

Dispone el VDP para una operación de lectura. Es mejor pasarla por alto y llamar directamente a la rutina situada en &H59.

### Posición: &H53

Prepara el VDP para una operación de escritura. Al igual que la anterior es mejor olvidarla y acceder a la rutina colocada en &H5C.

### Posición: &H56

Esta rutina llena la RAM de vídeo de un mismo valor contenido en el acumulador. La posición de origen debe encontrarse en HL y la longitud del bloque en BC. Modifica los pares AF y BC. La utilidad de esta rutina es colorear la pantalla rápidamente. Las instrucciones CLS, COLOR, LINE y PAINT la emplean.

### Posición: &H59

Esta rutina traslada un bloque de la RAM del VDP hacia la memoria central. La longitud del referido bloque ha de encontrarse en BC, el destino en DE y el origen en HL. Modifica AF, BC y DE. Tarde o temprano todos los programadores han de encontrarse con esta rutina, por lo que su uso es prácticamente imprescindible.

### Posición: &H5C

La rutina situada en esta dirección tiene un comportamiento análogo a la anterior, con la diferencia de que traslada un bloque desde la memoria central a la RAM de vídeo.

### Posición: &H5F

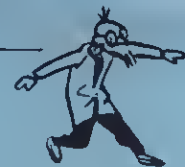
Esta llamada pone al VDP en uno de los cuatro modos de pantalla. El acumulador deberá contener el modo seleccionado. Su equivalente en BASIC sería SCREEN A. No inicializa los SPRITES. Modifica todos los registros así como las posiciones de memoria &HF3B0, &HF922, &HF924, &HFC4F y &HFCB0.

### Posición: &H62

Esta rutina cambia el color de la pantalla, tomando como nuevos valores las posiciones de memoria siguientes: &HF3E9 (color de la tinta), &HF3EA (color del papel) y &HF3EB (color del borde). Modifica los pares AF, BC y HL.

### Posición: &H89

Su cometido es inicializar todos los SPRITES. Altera todos los registros.



13, (A+8, B+32), 8, 21



#### **Posición: &H6C**

Esta rutina actúa como la instrucción BASIC SCREEN 0. Modifica todos los registros así como las posiciones de memoria que van desde la &HF3DF a la &HF3E5.

#### **Posición: &H6F**

Funciona igual que la anterior pero para el SCREEN 1.

#### **Posición: &H72**

Igual que las anteriores pero para SCREEN 2.

#### **Posición: &H75**

Igual para SCREEN 3.

#### **Posición: &H78**

Inicializa al VDP para trabajar en SCREEN 0, pero sin tocar la RAM de vídeo. Modifica los mismos registros y posiciones de memoria que la rutina situada en &H6C.

#### **Posición: &H7B**

Trabaja igual que la anterior pero para SCREEN 1.

#### **Posición: &H7E**

Igual que las anteriores pero para SCREEN 2.

#### **Posición: &H81**

Lo mismo para SCREEN 3.

#### **Posición: &H87**

Con esta rutina sólo tendrás que cargar un número de SPRITE en el acumulador para que te devuelva la dirección de la VRAM en la que se encuentran los atributos del SPRITE seleccionado, gracias al registro HL. Modifica los pares HL y DE así como los flags.

#### **Posición: &H8A**

Esta rutina te informará del tipo de SPRITE que estás empleando, o mejor dicho: el número de bytes que emplea cada uno de éstos, que pueden ser 8 ó 32. Por tanto, a la salida tendrás en el acumulador una de estas dos cantidades. Además el carry se pondrá a 1 si los SPRITES son del tipo ampliado. Únicamente modifica el par AF.

#### **Posición: &H8D**

Esta rutina escribe el carácter contenido en el acumulador en la dirección especificada por el cursor gráfico (la coordenada X está en &HF3B3 y la Y en &HF3B4), siempre y cuando estés trabajando en SCREEN 2. Sólo modifica las posiciones de memoria &HF92A, &HF923 y &HF92C.

#### **Posición: &H90**

Esta rutina inicializa el Generador Programable de Sonido. No modifica ningún registro, pero altera toda el área de la cola del sonido, que empieza en &HF975 y termina en &HFA74.

#### **Posición: &H93**

Con ella puedes escribir en uno de los registros del PSG. El número de registro ha de colocarse en el acumulador y en E el dato a mandar (comprendido entre 0 y 13). Su equivalente en BASIC sería: SOUND A, E. Esta llamada no modifica ningún registro.

#### **Posición: &H96**

Esta rutina sirve para leer un registro del PSG. El acumulador debe contener el número de registro (comprendido en 0 y 13). Sólo altera el contenido de A.

#### **Posición: &H99**

Se llama a esta rutina para ejecutar la escala musical (caso de haberla). Si en el buffer de sonido no hay ninguna escala escrita el acumulador se cargará con un cero. Modifica los pares AF y HL, así como las posiciones de memoria &HFB3F y &HFB40.

#### **Posición: &H9C**

Comprueba si las teclas de función están activas en la pantalla. En caso afirmativo, analiza las teclas SHIFT, para mostrar el contenido de las funciones F6 y F10, si están pulsadas. Esta rutina pondrá el flag Z a 1 si no hay ninguna tecla apretada. Únicamente modifica AF.

#### **Posición: &H9F**

Esta rutina es de gran importancia. Su cometido es coger un carácter del buffer del teclado. Si este buffer está vacío enseñará el cursor y esperará hasta que se pulse una tecla. A la salida, el acumulador contendrá el código del carácter. Asimismo, realiza una llamada al gancho situado en &HFDC2 después de apilar los pares HL, DE y BC. No modifica ningún registro.

#### **Posición: &HA2**

Imprime el carácter del acumulador en la posi-





ción en la que se encuentre el cursor, aunque se trate de un código de control. Actualiza la pantalla, desplazándola o haciendo un cambio de línea si es preciso. Después de apilar todos los registros salta al gancho situado en &HFDA4. No modifica ningún registro pero sí las coordenadas Y y X del cursor (almacenadas en &HF3DC y &HF3DD respectivamente) y la dirección &HF661.

#### **Posición: &HA5**

Envía el carácter contenido en el acumulador a la impresora, esperando hasta que ésta esté preparada. Si se pulsa CTRL-STOP el flag C se pondrá a 1. No modifica ningún registro.

#### **Posición: &HA8**

Esta rutina es llamada por la anterior. Su finalidad es comprobar si la impresora está ON-LINE. De no ser así el flag Z se pondrá a 1. Modifica el par AF.

#### **Posición: &HAB**

Transforma el código contenido en el acumulador en un carácter gráfico (si es menor que 32), en la forma que el VDP está preparado para aceptar. Prueba con VPOKE 0,1 y entenderás perfectamente el funcionamiento de esta rutina. Modifica el par AF.

#### **Posición: &HAE**

Acepta una línea completa del teclado. Puesto que una línea puede contener hasta 255 caracteres, ésta se almacena en buffer de entrada que está situado entre las posiciones &HF55E y &HF65D. A la salida, el par HL apunta al inicio de este buffer menos uno. Modifica todos los registros.

#### **Posición: &HB1**

Esta rutina es similar a la anterior. Aceptará la entrada de caracteres e irá mostrándolos en la pantalla hasta que se pulse RETURN o CTRL-STOP. Modifica todos los registros.

#### **Posición: &HB4**

Esta rutina actúa de forma idéntica a las anteriores, pero visualizando antes el signo de interrogación característico de los INPUT.

#### **Posición: &HB7**

Sirve para comprobar si se ha pulsado CTRL-STOP. Si esto es así, el flag C se pondrá a 1. Modifica AF.

#### **Posición: &HBA**

Esta rutina complementa a la anterior, pero además analiza si se ha pulsado únicamente la tecla STOP, para detener la ejecución del programa cuando así sea. Altera el par AF.

#### **Posición: &HBD**

Esta rutina hace exactamente lo mismo que la anterior, pero empleando más tiempo.

#### **Posición: &HCO**

Produce un BEEP e inicializa el PSG, llamando a la rutina situada en &H90. Modifica todos los registros. Su equivalente en BASIC sería: BEEP.

#### **Posición: &HC3**

Su cometido es borrar la pantalla, con la condición de que pongas el flag Z a 0 antes de llamarla. Modifica los pares AF, BC y DE y las posiciones de la RAM del sistema relacionadas con el cursor. El modo de pantalla que se esté utilizando es indiferente.

#### **Posición: &HC6**

Sitúa el cursor en la posición especificada por el registro HL, para lo cual es necesario poner la columna en H y la fila en L. Altera el par AF y las direcciones de memoria encargadas de guardar las coordenadas de cursor (&HF3DC y &F3DD). Su equivalente en BASIC sería: LOCATE L, H.

#### **Posición: &HC9**

Esta rutina es llamada por el intérprete BASIC para saber si las teclas de función están activas.

#### **Posición: &HCC**

Se llama a esta rutina para desconectar la visualización de las teclas de función. Su equivalente en BASIC sería: KEYOFF. Altera AF, BC y DE.

#### **Posición: &HCF**

Puede utilizarse para mostrar el contenido de las teclas de función en la pantalla. Actúa como la instrucción BASIC KEYON. Modifica los registros AF, BC y DE, así como la posición &HF3DE, que será cargada con &HFF.

#### **Posición: &HD2**

Esta rutina se emplea para cambiar de pantalla y ponerla en el otro modo de texto.

#### **Posición: &HD5**

Esta llamada realiza una función idéntica a la instrucción BASIC A=STICK(A), por lo que te sugiero que leas el manual de tu ordenador para conocer los detalles. Modifica todos los registros.

#### **Posición: &HD8**

Analiza el estado del disparador especificado por un número que debe cargarse en el acumulador. A la salida, tendrás un cero en el registro A, si ha habido algún disparo, o 255, si no se ha pulsado el disparador. Modifica AF.

#### **Posición: &HDB**

Esta rutina funciona de forma análoga a la instrucción BASIC PAD(A). Por consiguiente, te aconsejo que mires allí para obtener una información completa. Altera todos los registros.

#### **Posición: &HDE**

Esta rutina lee la raqueta de juegos especificada por el registro A. Asimismo, devuelve en el acumulador un parámetro comprendido entre 0 y 255, referido a la posición actual. Modifica todos los registros.

#### **Posición: &HE1**

Con esta llamada pondrás el motor del cassette en marcha y podrás leer la cinta. Si se pulsa CTRL-STOP el flag C se pondrá a 1. Modifica todos



los registros.

**Posición: &HE4**

Se emplea para leer un byte de la cinta, que será devuelto en el acumulador. Al igual que la rutina anterior, el carry se encenderá si la operación es abortada. Modifica todos los registros.

**Posición: &HE7**

Esta rutina sirve para detener la operación de lectura del cassette. No altera ningún registro.

**Posición: &HEA**

Esta rutina pone el motor del cassette en marcha y escribe la cabecera en la cinta. El carry se pondrá a 1 si se interrumpe la escritura. Modifica todos los registros.

**Posición: &HED**

Carga el acumulador con un dato y esta rutina te lo escribirá en la cinta. Como siempre el carry encendido te indicará si la operación fue abortada por la pulsación de CTRL-STOP. Modifica todos los registros.

**Posición: &HF3**

Esta rutina conectará el motor del cassette, si el acumulador contiene un 1, o lo parará, si contiene un 0. Por otra parte, si cargas el registro A con &HFF, antes de llamarla, invertirás el estado del motor.

**Posición: &HFC**

Esta rutina desplaza al cursor gráfico un punto hacia la derecha. Al llamarla, la posición &HF92A y siguiente debe contener la dirección de la VRAM en la que se encuentra el punto. Asimismo, deberás po-

ner la en posición &HF92C un valor cuyo único bit encendido muestre el punto a tratar. Por consiguiente si &HF92C contiene un 32 (&B00001000) el cursor gráfico señalará al tercer punto de la posición especificada por &HF92A, al volver de la rutina. Modifica el par AF y las tres posiciones de memoria antes referidas.

**Posición: V &HFF**

Esta rutina hace exactamente lo mismo que la anterior, sólo que el cursor gráfico se desplaza un punto a la izquierda.

**Posición: &H102**

Hace lo mismo que las anteriores pero desplazando el cursor hacia arriba.

**Posición: &H105**

Trabaja igual que la rutina anterior pero pone el carry a 1 si se alcanza la fila superior de la pantalla.

**Posición: &H108**

Se comporta como &HFC pero bajando un punto el cursor gráfico.

**Posición: &H10B**

También hace bajar un punto el cursor gráfico, aunque pondrá el carry a 1 si se llega a la fila inferior de la pantalla. El resto como &HFC.

**Posición: &H11D**

Esta rutina devuelve en el acumulador el código de color del punto señalado por las posiciones de memoria &HF92A a &HF92C (ver la rutina situada en &HFC).

**Posición: &H123**

Esta rutina traza una línea hacia la derecha a partir de la posición especificada por las direcciones &HF92A a &HF92C (ver la rutina situada en &HFC) y la longitud contenida en HL. El color del trazo ha de colocarse en &HF3F2. Modifica todos los registros.

**Posición: &H132**

Usando esta rutina actuarás directamente sobre el diodo de las mayúsculas. Así, si el acumulador contiene un cero lo encenderás, con otro valor, lo apagarás. Modifica el par AF.

**Posición: &H141**


Esta rutina comprueba el estado de la matriz del teclado. Dicha matriz forma un cuadrado de 8x8. El acumulador deberá contener el número de la fila a explotar. A la salida tendrás que A tiene un 255, si no ha sido pulsada ninguna tecla de la fila en cuestión, o un bit puesto a cero, indicando la tecla que sí se ha pulsado. Únicamente altera el par AF y no espera hasta que se pulsa una tecla.

**Posición: &156**

Sirve para borrar completamente el buffer del teclado. Modifica el registro HL.

**Nota:** Las posiciones de la ROM 6 y 7 contienen los números de los puertos asignados para las operaciones de entrada/salida al VDP.



  
 X(W) AND X  
 THEN 960 AND  
 420 NEXT W: RETURN

## VARIABLES ROM DEL SISTEMA

### DIRECCION

#### FUNCION

- 003E Inicializar teclas funcionales.  
MODIFICA Todos los registros.
- 004A Leer datos de la VRAM  
ENTRADA HL: dirección VRAM  
SALIDA A: datos  
MODIFICA AF
- 004D Escribir datos en la VRAM  
ENTRADA HL: dirección VRAM  
A: datos  
MODIFICA AF
- 0056 Introducir una constante en la VRAM  
ENTRADA BC: longitud  
HL: dirección VRAM  
A: datos  
MODIFICA AF, BC
- 0059 Transferir un bloque de la memoria principal a la VRAM  
ENTRADA BC: longitud  
DE: dirección RAM de destino  
HL: dirección VRAM de origen  
MODIFICA Todos los datos
- 005C Transferir un bloque de la memoria principal a la VRAM  
ENTRADA BC: longitud  
DE: dirección VRAM de destino  
HL: dirección RAM de origen  
MODIFICA Todos los registros
- 0090 Inicializar el generador programable de sonidos (PSO)  
MODIFICA Todos los registros
- 0093 Escribir datos en el PSG  
ENTRADA A: n.º del registro
- 0096 Leer datos del PSG  
ENTRADA A: n.º de registro  
SALIDA A: datos  
MODIFICA A
- 009C Verificar buffer de teclado  
SALIDA Cero (flag) si el buffer está vacío
- 009F Esperar una entrada de teclado  
SALIDA A: el carácter  
MODIFICA AF
- 00D5 Examinar estado del joystick  
ENTRADA A: etick ID (0-2)  
SALIDA A: etick status (0-8)  
MODIFICA Todos los registros
- 00D6 Examinar disparador  
ENTRADA A: disparador ID (0-4)  
SALIDA A: 255 si está pulsado  
MODIFICA AF
- 0141 Obtener el estado de la matriz del

teclado

ENTRADA A: dirección de la fila

SALIDA A: estado de la fila

MODIFICA AF

0166 Borrar buffer de teclado

MODIFICA HL





## H. VARIABLES RAM DEL SISTEMA

### DIRECCION

#### FUNCION

F360 rutina para leer la ranura primaria  
 F365 rutina para escribir en la ranura primaria  
 F36C llamar rutina de la ranura primaria  
 F39A dirección inicial para USRO-9  
 F3AE longitud de línea = 39  
 F3AF longitud de línea = 31  
 F3BO longitud de línea  
 F3B1 líneas en pantalla = 24  
 F3B2 espacio de columna = 14  
 F3B3 SCREEN 0 tabla de nombres  
 F3B5 tabla de colores  
 F3B7 forma de carácter  
 F3B9 atributo  
 F3BB sprite  
 F3BD SCREEN 1 tabla de nombres  
 F3BF tabla de colores  
 F3C1 forma de carácter  
 F3C3 atributo  
 F3C5 sprite  
 F3C7 SCREEN 2 tabla de nombres  
 F3C9 tabla de colores  
 F3CB forma de carácter  
 F3CD atributo  
 F3CF sprite  
 F3D1 SCREEN 3 Tabla de nombres  
 F3D3 tabla de colores  
 F3D5 forma de carácter  
 F3D7 atributo  
 F3D9 sprite  
 F3DB enganche de tecla  
 F3DC coord. Y cureor  
 F3DD coord. X cursor  
 F3DE teclas funcionales  
 F3DF contenido del registro VDP  
 F3E7 = 0  
 F3E8 = (FF)  
 F3E9 color de primer plano  
 F3EA color de fondo  
 F3EB color de borde  
 F3EC salto 0  
 F3EF salto 0  
 F3F2 byte atributo  
 F3F3 dirección de tabla de espera  
 F3F5 = (FF)  
 F3F6 sincronización de exploración de teclas  
 F3F7 = 50  
 F3F8 (put) buffer teclado  
 F3FA (get) buffer teclado  
 F3FC parámetros de E/S cassette  
 F40F puntero de RESUME TEXT  
 F414 código de error  
 F416 cabeza impresora  
 F416 salida impresora  
 F417 0 = para impresora MSX  
 F416 distinto de cero para salida de caracteres sin procesar

F419 función val  
 F41C línea cursor  
 F41F buffer de proceso  
 F55D coma para INPUT  
 F55E buffer de entrada de teclado  
 F660 fin de buffer  
 F661 posición terminal  
 F662 flag de matriz  
 F663 tipo de valor  
 F664 tipo de operador  
 F665 para proceso  
 F666 puntero de texto para gschr  
 F666 forma interna de la constante posterior a gschr  
 F669 tipo de constante  
 F672 parte superior de la memoria  
 F674 parte superior de la pila  
 F676 parte superior del texto  
 F676 descripción temporal  
 F67A almacenar descripciones temporales  
 F696 descripción de cadena después de operaciones  
 F69B parte superior posible del espacio de cadenas  
 F66D para operaciones de reorganización de datos  
 F6A1 puntero de sentencia FOR  
 F6A3 puntero de sentencia DATA  
 F6A5 flag para FOR Y USR  
 F6A6 flag para INPUT Y READ  
 F6A7 para sentencias  
 F6A9 = O cuando no hay línea de programa  
 F6AA = 0 en modo AUTO  
 F6AD incremento en AUTO  
 F6AF puntero de texto para RESUME  
 F6B1 grabar pila para proceso de errores  
 F6B3 línea de error  
 F6B5 línea de curso  
 F6B7 puntero de texto para RESUME  
 F6B9 línea de proceso de errores  
 F6BE = 1 si se está procesando un error  
 F6BC tarsas temporales  
 F6B6 antiguo n.º de línea establecido por CTRLSTOP, STOP Y END  
 F6CO antiguo puntero de texto  
 F6C2 dirección inicial de variables simples  
 F6C4 dirección inicial de matrices  
 F6C8 fin de la memoria utilizada  
 F6C8 puntero DATA  
 F6CA tipo de variable para A-Z  
 F6E4 pila usada en labores de recogida de basura  
 F6E6 longitud de tabla  
 F6E6 tablas de parámetros para funciones definidas para el usuario  
 F74C puntero de bloqueo de parámetros  
 F74E longitud del bloqueo de parámetros  
 F750 direcciones de los parámetros

F7B4 flag para búsqueda de parámetros  
 F7B5 fin de búsqueda  
 F7B7 = 0 si no corresponde función  
 F7BA uso temporal en recogida de basura  
 F7BC para uso de intercambios  
 F7C4 = 0 para rastreo desactivado  
 F7C5 = zona de trabajo para rutinas de paquetes BCD  
 F63F = zona de datos para manipulación de ficheros  
 F67F contenido de teclas funcionales  
 F91F tablas de VRAM BASE  
 F92A para GENGRP  
 F931 zona de trabajo y CIRCLE  
 F949 zona de trabajo de PAINT  
 F955 zona de trabajo de PLAY  
 FBBO posible recalentamiento si es distinto de cero  
 FBB1 distinto de cero el texto BASIC está en ROM  
 FBB2 tabla de terminadores de línea  
 FBCA primera posición de carácter en INLIN  
 FBCC código para cursor  
 FBCE flag para teclas funcionales  
 FBCE flags para interruptores condicionales por teclas de función  
 FBD6 flag de condición  
 FBD9 flag de enganche  
 FBDA antiguo estado de tecla  
 FBE5 nuevo estado de tecla  
 FBFO buffer de código de tecla  
 FC16 operaciones de proceso de pantalla  
 FC40 operación de pattern converter  
 FC46 parte inferior de la RAM  
 FC4A parte superior de la memoria  
 FC4C tabla de interrupción  
 FC9A RTYCNT  
 FC9B INTFLG  
 FC9C PAD X  
 FC9D PAD Y  
 FC9E JIFFY  
 FCA0 intervalo  
 FCA2 contador de intervalo  
 FCA4 leer cassette  
 FCA6 encabezamiento de carácter gráfico  
 FCA7 contador de secuencia de escape  
 FCAB flag de inserción  
 FCA9 ON/OFF cureor  
 FCAA carácter de cureor  
 FCAB estado de la tecla CAPS  
 FCAC operaciones de la tecla desactivada  
 FCAD no utilizada  
 FCAE = 0 mientras se carga un programa BASIC  
 FCAF modo de pantalla (screen)  
 FCBO antiguo modo ecresn  
 FCB1 carácter para CAS:  
 FCB2 color de borde en PAINT  
 FCB3 cursor gráfico, coord. X  
 FCB5 cursor gráfico, coord. Y  
 FCB7 acumulador gráfico, X  
 FCB9 acumulador gráfico, Y  
 FCB8 flag de DRAW  
 FCB8 escala en DRAW  
 FCBD ángulo de DRAW  
 FCBE BLOAD/BSAVE  
 FCBF inicio de BSAVE  
 FCCI zona de trabajo de ranura  
 FD9A enganches